

# Velocity based controllers for dynamic character animation

Xavier Lligadas	Antonio Susin	Jorge E. Ramirez
Dept. de matemàtica aplicada 1	Dept. de matemàtica aplicada 1	Dept.de Llenguatges i sistemes informàtics
Universitat Politècnica de Cataluya (UPC)	Universitat Politècnica de Cataluya (UPC)	Universitat Politècnica de Cataluya (UPC)
Barcelona, Spain	Barcelona, Spain	Barcelona, Spain
xavier.lligadas@upc.edu	toni.susin@upc.edu	jramirez@lsi.upc.edu

## Abstract

Dynamic character animation is a technique used to create character movements based on physics laws. Proportional derivative (PD) controllers are one of the preferred techniques in real time character simulations for driving the state of the character from its current state to a new target-state. In this paper is presented an alternative approach named velocity based controllers that are able to introduce into the dynamical system desired limbs relative velocities as constraints. As a result, the presented technique takes into account all the dynamical system to calculate the forces that transform our character from its current state to the target-state. This technique allows real-time simulation, uses a straightforward parameterization for the character muscle force capabilities and it is robust to disturbances. The paper shows the controllers capabilities for the case of human gait animation.

## 1 Introduction

3D character animation is a very complicated field due to the innate capacity of people to differentiate between human natural movements and unnatural ones.

Techniques as motion capture, produce good-looking natural movements but they fail when trying to scale them to a simulated environment where a great number of movements

could be needed. Generation of different walking styles, which are dependent on a desired velocity or the possible variations of the terrain, is a difficult task to fulfill without the use of large amounts of motion capture data.

In order to generate animations able to adapt itself to different environments, a more algorithmic approach is mandatory. Physically-based animation is an example of an algorithmic approach to the problem. Using physics laws we can generate dynamically consistent animations which can adapt itself to changing environments.

When dynamics are used to animate a character, one of the main problems we must face is controlling the limbs of the character. In the graphics community we find optimization based methods and controllers based methods to drive forward dynamics simulations. Usually, optimization based methods [27],[22] obtain better results than controllers based methods, but in real time environments they are not appropriate because of their time consumption and their possible unstabilities. On the other hand, controllers based methods [35],[34] have the advantage of the performance and the capability of using feedback laws to adapt and interact with the environment. Both methods can be used with motion capture animation to obtain realistic movements capable to adapt itself to the environment.

Usually, controllers based methods [34],[33]

drive the character toward target-states created by an animator or introduced to the system by a motion capture file. At each time-step the controllers must calculate the forces and torques needed in every limb of the model to transform the model from its current state to the target-state.

In this paper is presented a controllers based method that instead of setting the PD torque directly into the dynamical system, it calculates the relative desired velocities between limbs using the current state and the target-state that our character must achieve. The computed velocities are then introduced as constraints into the dynamical system. The constrained system obtained can be defined as a linear complementarity problem (LCP) which we solve using the Dantzing's algorithm implemented in the physics engine ODE [23]. Finally, the system executes a forward simulation step with the forces and torques obtained from the LCP as input parameters in charge of driving the character from its current state to the target-state in a given optimal time. This approach based on velocity transitions between states has made the overall system more intuitive from a user point of view.

## 2 Previous Work

In [17] it is presented a method capable to create different human-like behaviors using kinematics constraints and dynamic equations.

A more specific work [6] presents an algorithmic approach which is able to generate walking and running applications based on the dynamics of an inverted pendulum with a telescopic joint for the stance leg, and a double pendulum model for the swing leg.

Two main approaches arise in the community in order to generate dynamic character animation, optimization based methods [32] and controllers based methods [26].

Real time optimization techniques used motion capture data as an initial solution, and try to solve the problem transforming it into a low-dimensional space problem [27],[8], [25],[12] or using some offline pre-calculation [22] that is used later in real-time execution. The main

problems related to these methods is that they can find erroneous solutions given by local minima, and although real-time capable, they usually require all the computer processor time for one character simulation.

In [15],[16] and [31] it is introduced finite state machines to specify cyclic motions which are driven using PD controllers. They are suitable for simple animations but they lack from an extensible balance algorithm. Another finite state machine method that uses PD controllers [34] shows a method for creating different locomotion skills with a small set of target-states, and a global center of mass based balance strategy. Another form of PD controllers based balance control is showed for cyclic animations in [20].

In [11] it is proposed a framework based on controllers where controllers created by the community can be used together. Although not necessary, the controllers it proposes are PD based.

PD controllers have been also used with motion capture animations. [14] adapts animations captured for a specific actor to other actors with different physical characteristics.

In [3],[35],[36], and [30] it is used PD controllers capable of reacting naturally to pushes and fallings in the same way as [10]. The work in [33] introduces a PD controller system that following motion captures global orientations, and introducing a fictional force acting over the root, the system is capable to adapt motion capture animations to simulated environments as the ones used in videogames or virtual reality applications.

Others methods [7],[9], and [1] try to mix optimization with controllers methods. In these techniques, the optimization process makes the animation aware of future changes in the simulated environment, and the controller is used to react instantly to unexpected variations and to decrease posible tracking errors. Another way of mixing these techniques is showed in [29] where an optimization process is used to train PD controllers.

A different real time approach to character animation is showed in [21] where simple dynamics are used with keyframed animation to ob-

tain simulated character behaviors.

### 3 Velocity based controllers

The controllers developed in this paper calculate the forces that act on the system based on the necessary velocity for the model to achieve the target-state in an optimal time.

This controllers can be built using any physics engine; we have chosen ODE [23], a physics engine aimed to real time simulations that uses a variation of the Dantzing's algorithm [5] to calculate its constraints.

As any dynamic system driven by controllers we need some joint target-states, in our case the target-states are introduced by an animator (see Fig. 2).

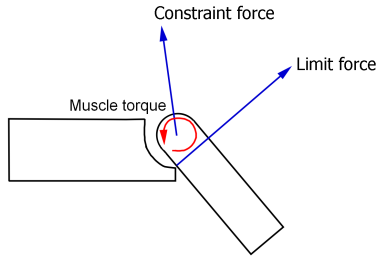


Figure 1: Forces exerted in a joint.

#### 3.1 Joints

Apart from the root, only rotational joints are used in our character model (see Fig. 7), therefore, the state of an object relative to its parent can be represented by a quaternion [28], which is easily used to calculate the desired velocity at each time step.

Our solver classifies the forces that a joint exerts as is shown in Fig. 1):

- **Constraint force:** Forces introduced by the geometry of the joint, ball-and-socket joint 3 Degrees of freedom (Dof), universal joints 2Dof, and hinge joint 1Dof.
- **Limit force:** These forces must restrict the orientation of the limbs relative to their

parents. Simple Euler's restrictions implemented in ODE have been used for the Universal and the Hinge joints. For the ball-and-socket joint we have used the work in [4] to create shoulder and hip like articulation limits.

- **Muscle torque:** The torque done by the group of muscles acting over the joint.

Our controllers only must calculate the muscle torque necessary to reach the desired velocity that will drive our system to the target-state in the optimal time specified by the animator.

#### 3.2 Calculation of desired Velocity

Given the current quaternion of a limb  $q_c$  and the target-quaternion  $q_d$  introduced by an animator, the system is able to calculate the quaternion that transforms the limb from its current state  $q_c$  to its desired state  $q_d$ . The transformation quaternion from  $q_c$  to  $q_d$  is calculated with a simple multiplication  $q_t = q_c^* * q_d$  where  $q_c^*$  is the quaternion conjugate. As any quaternion can be transformed into an axis-angle rotation [28], the system can obtain from  $q_t$  its axis of rotation  $v_t$  and its angle of rotation  $\theta_t$ . Finally, with the given optimal time  $t$  introduced by the animator (see Fig. 2), the system is able to calculate the relative angular velocity  $w = \theta_t * v_t / t$  that the models limbs must achieve to transform its current state to the target-state in the given optimal time.

This is the velocity that our controllers will try to reach. As this is a dynamic simulation, our solver will calculate the acceleration that our dynamical system needs to achieve this velocity. This can be done using any inverse dynamics algorithm [13], in our case we have used the Dantzing's algorithm implemented in ODE.

In order to simulate human muscle activation, two parameters are used in our system. The maximum torque that a muscle is able to exert  $\tau_{max}$  and the muscle reaction time  $t_r$  (see table 2). Then at every time-step, the maximum torque that a muscle will be able to exert is going to be  $\tau = (\tau_{max} / t_r) * \Delta t + \tau_{prev}$  being  $\Delta t$  the time-step, and  $\tau_{prev}$  the torque

applied in the previous simulated step. We have found these two parameters much easier to tune than the gains of a proportional derivative controller, it is much more intuitive to tune the reaction time of a muscle than the damping constant of a PD controller.

#### 4 Animation system

To test the controllers developed, a finite state machine (FSM) system as the one presented in [34] has been created. The target states used the FSM are closest to the real desired states than the ones used in [34]. Therefore, the system is easier to tune.

The presented system is aimed to gait generation, that is why the character model has been represented by a trunk, a hip (root), and the legs (see Fig. 7). In Figure 2 it is shown the states used for a walking animation and the time or events (foot strikes) used to change from one target-state to another.

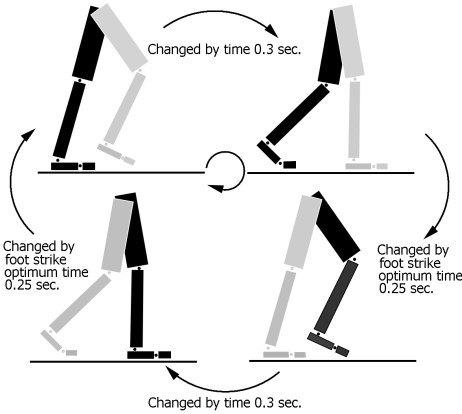


Figure 2: Finite state machine used for our walking animation.

##### 4.1 Combining Controllers with joint target-positions

The main problem to combine the controllers with the character target-positions is the fact that no muscle actuate directly over the root (see Fig. 3). The total torque acting over

the root is the sum of all the torques used to control its limbs with inverted sign.

As the controllers developed calculate velocities based on relative orientations between bodies, they do not have knowledge of the global orientation. For the controllers developed in this work, is the same transforming the models current state to the state (Fig. 4) *state (a)* than to the the state (Fig. 4) *state (b)* because the relative orientations between limbs are similar.

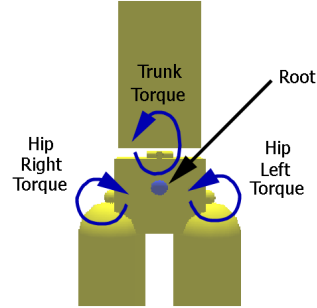


Figure 3: Root forces.

To solve this problem the system uses an additional joint for the model. Each time a foot contacts with the ground, the system creates an additional joint in order to fix the contacting foot to the ground. The additional joint prevents foot-sliding, and give knowledge to the system about its global orientation. Using this strategy the controllers will be able to drive the character toward (Fig. 4) *state (b)* instead of (Fig. 4) *state (a)*. With this change every limb of the character is actuated and now the stance thigh joint is in charge to control the orientation of the hip.

##### 4.2 Constrained environment

With a given foot position and 2D inverse kinematics the character is able to walk in a constrained environment as [7] without requiring previous optimization.

The inverse kinematics proposed is done solving the triangle shown in (Figure 5). Given a

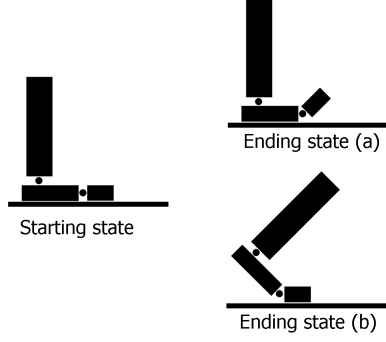


Figure 4: States relative orientation.

foot position and some constraints on the orientations of the stance foot, toe, calf and swing foot at ground strike, the system is able to calculate the step distance needed  $d$  (see Fig. 5) and use the equations 1 to obtain a new target-position for the double support states of the animation system (see Fig. 2).

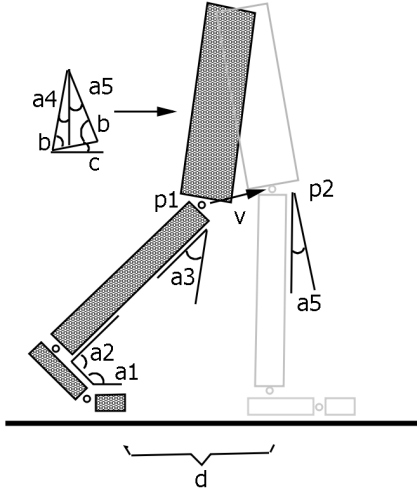


Figure 5: Inverse kinematics triangle.

$$\begin{aligned}
 a_1 &= \frac{\pi}{4a_2} = \frac{\pi}{2.0} \\
 l &= \text{mod}(v) \\
 b &= \text{acos}\left(\frac{l}{2.0} * \text{thigh}_{length}\right) \\
 c &= \text{atan}\left(\frac{v_y}{v_x}\right) \\
 a_4 &= \frac{\pi}{2.0} - b - c \\
 a_5 &= b - c - \frac{\pi}{2.0} \\
 a_3 &= \frac{\pi}{4.0} - a_4
 \end{aligned} \tag{1}$$

If the desired position imposed by the constrained environment is unreachable for the model, the inverse kinematics method proposed will try to find the closest state to the desired foot position.

### 4.3 Balance

Using the constrained environment method, a balance strategy based on the character base polygon [24] has been developed.

The system is able to control the position of the next foot strike, and varying it the system can transform the base polygon by positioning the next foot strike to a different distance from the current stance foot.

As shows (Figure 6) the balance strategy proposed is activated when the projection of the character center of mass over the base polygon is beyond the 25% of the base polygon center (outside the tolerance polygon). If balance is necessary to maintain equilibrium the new foot position is recalculated as  $d = 4(cCM - bP)/3 - 2\text{foot}_{length}$  (for an explanation of parameters see Fig. 6). With this new foot position the system is positioning the center of mass further from the zero moment point [19], therefore increasing the balance of the character.

## 5 Results

The simulations only consume 10% of the CPU computation time on a 1.66Ghz processor with 1GB RAM under XP operating

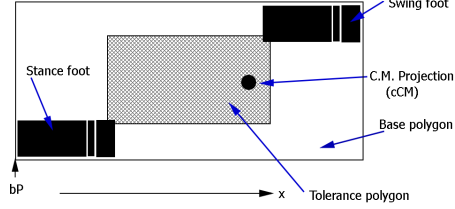


Figure 6: Balance polygon.

<i>Limb</i>	<i>Length (cm)</i>	<i>Mass (Kg)</i>
Trunk	40.9	40
Hip	21.09	16.3
Thigh	45.85	8.3
Calf	45.05	4.04
Foot	19	0.86
Toe	6.1	0.15

Table 1: Limb parameters

system.

The physics are simulated with ODE physics engine using a time-step of 0.005s.

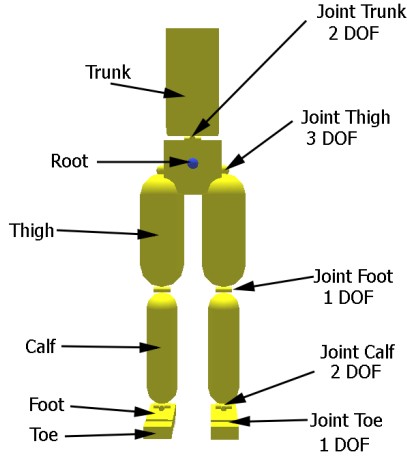


Figure 7: Character Model.

### 5.1 Model

The system uses a simplified model of the human body with the purpose of gait simulation. The proposed model has 24 degrees of freedom (Figure 7). Table 1 shows the parameters used for the limbs of the model, and table 2 for the muscles of a human male of 179.9cm height and 82,2 Kg. For models with different height or weight the parameters are calculated proportionally. The anthropometric distances were extracted from the NASA study [2]. The

maximum torque that a muscle can exert is based on experimentation and the studies done by the NASA. The muscles reaction time has been set to 20 ms for all the muscles. The system presents a stability problem if reaction times were superior to 20 ms in some muscles. This is a problem we will try to solve in the future because from the biomechanics community [18] can be extracted that 100 ms is an average time for a muscle to exert the 80% of its maximum force which is something we want to achieve in our simulations.

<i>Joint</i>	<i>DOF</i>	<i>Torque (Nm)</i>
TrunkJoint	2	(600, 0, 300)
ThighJoint	3	(500, 500, 500)
CalfJoint	1	(600, 0, 0)
FootJoint	2	(300, 0, 200)
ToeJoint	1	(50, 0, 0)

Table 2: Joint parameters

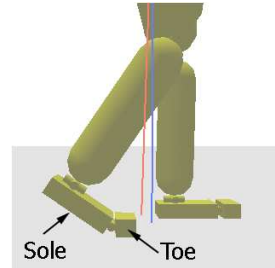


Figure 8: Detail of toe movement in stance foot load balance phase.

## 5.2 Walking

The controllers proposed are able to animate a system based on muscular forces. The movement is still robotic but the high stability of the controllers has shown a lot of margin for improvement. For instance, the system is reproducing the toe movement in our walking simulations (Figure 8), something that we have never seen before in any dynamically driven character animation due to its big instability.

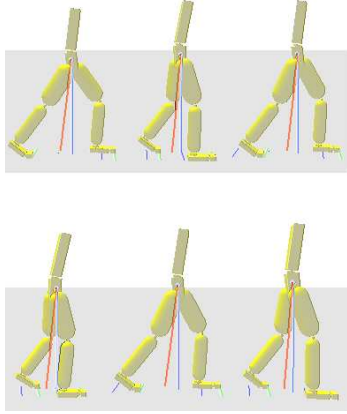


Figure 9: Constrained environment forcing changes in the step length. Green and red lines mark starting and ending positions.

## 5.3 Constrained environment

Here we show how the system is capable to walk in an environment constraining its foot steps to a certain distance. In (Figure 9) we show a steps sequence of 0.60 meters, 0.20 meters, 0.50 meters, 0.16 meters, 0.50 meters and 0.30 meters, showing how the system is capable to change from big steps to small ones continuously.

We have noticed an error increase for very large and very small steps. In the first case is due to the foot controllers not being able to reach the joint-target state although being near. In the second case is due to the restriction the system sets on the foot stance leg, the

calf, and the foot orientations in the inverse kinematics method.

## 5.4 Reacting to pushes

With the proposed balance control, the character is able to react to sagittal pushes in a range of (-250 , 600) Newtons during 0.5 seconds over the trunk (Figure 10) or the root. For pushes bigger than these, the character lose its balance and fall (Figure 11.)

We must say that due to some stability problems we could not set the muscle reactions to values as high as they are in real life, as a result the movements obtained in the simulations are less reactive than the obtained with PD controllers [34] because the force variation over time in the system proposed is higher than the obtained with PD controllers.

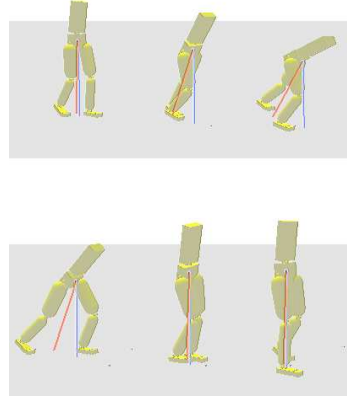


Figure 10: Character balancing a 500N push.

## 6 Conclusions

The paper has presented a new force calculation method for dynamically driven character animation. The method is easier to use than other proposed methods because the target positions of the animation system are more similar to the animations that the animator wants the character to perform.

The parameterization of the system only needs two intuitive parameters, the maximum torque of the joint, and the reaction time of the group of muscles acting over the joint.

The stability, reactivity and efficiency of our controllers have been shown in a walking simulation which can be used in a constrained environment.

The paper has presented a joint constrained method for foot contact which has made possible the simulation of some unstable biomechanical gait phases as toe-off.

The controllers presented are unstable if we try to simulate real muscle reactions, that is why the muscle reaction times are small and the controllers are not as reactive as the PD controllers.

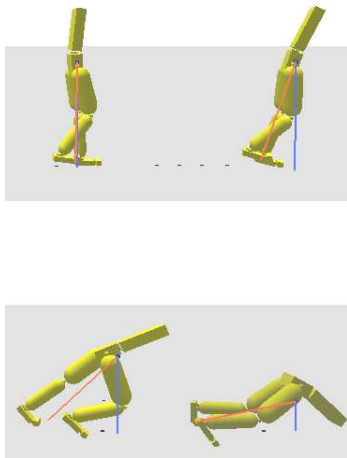


Figure 11: Character falling under a 700N push.

## 7 Future work

This is the initial framework we have created for dynamic character animation.

We plan to work on the stability of the muscle reaction to obtain more continuity in the velocity of the system.

Until now we have worked in the sagittal plane, the most important one for walking,

but some attention must be set into the frontal plane, something we are planning to do in the future.

To improve realism, we want to study the foot motion as an articulated structure, viewed as a movement of two limbs, we have started this line of research but there is still a lot of margin for improvements.

New controllers as running or jumping will be developed as well as an specialized balance controller based in our inverse kinematic method which could adapt the animation to different situations as transversal pushes, terrain variations or objects avoidance.

## 8 Acknowledgments

Partial support was provided by Spanish Grant: TIN2007-67982-C02-01, El comissionat per a universitats i recerca del departament d'innovació d'universitats i empreses de la Generalitat de Catalunya, and the European Social Fund.

Finally, we would like to thank the LABSID team for their interesting and valuable discussions.

## References

- [1] Yeuhi Abe, Marco da Silva, and Jovan Popović. Multiobjective control with frictional contacts. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 249–258, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.
- [2] NASA. Antropometry and biomechanics. <http://msis.jsc.nasa.gov/sections/section03.htm>.
- [3] Zordan V. B. and Jessica K. Hodgins. Tracking and modifying upper-body human motion data with dynamic simulation. In *In Computer Animation and Simulation '99*, 1999.
- [4] Paolo Baerlocher and Ronan Boulic. Parametrization and range of motion of



- the ball-and-socket joint. In *DEFORM '00/AVATARS '00: Proceedings of the IFIP TC5/WG5.10 DEFORM'2000 Workshop and AVATARS'2000 Workshop on Deformable Avatars*, pages 180–190, Deventer, The Netherlands, The Netherlands, 2001. Kluwer, B.V.
- [5] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 23–34, New York, NY, USA, 1994. ACM.
- [6] A. Bruderlin and T. W. Calvert. Goal-directed, dynamic animation of human walking. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 233–242, New York, NY, USA, 1989. ACM.
- [7] Stelian Coros, Philippe Beaudoin, Kang Kang Yin, and Michiel van de Pann. Synthesis of constrained walking skills. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–9, New York, NY, USA, 2008. ACM.
- [8] Marco da Silva, Yeuhi Abe, and Jovan Popović. Interactive simulation of stylized human locomotion. *ACM Trans. Graph.*, 27(3):1–10, 2008.
- [9] Marco da Silva, Yeuhi Abe, and Jovan Popović. Simulation of human motion data using short-horizon model-predictive control. In *Computer Graphics Forum 27*. In press, 2008.
- [10] Natural Motion. Endorphin, 2006. <http://www.naturalmotion.com>.
- [11] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 251–260, New York, NY, USA, 2001. ACM.
- [12] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. *ACM Trans. Graph.*, 22(3):417–426, 2003.
- [13] Roy Featherstone. *Robot Dynamics Algorithm*. Kluwer Academic Publishers, Norwell, MA, USA, 1987. Manufactured By-Publishers, Kluwer Academic.
- [14] Jessica K. Hodgins and Nancy S. Pollard. Physically realistic morphing. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997.
- [15] Jessica K. Hodgins, Paula K. Sweeney, and David G. Lawrence. Generating natural-looking motion for computer animation. In *Proceedings of the conference on Graphics interface '92*, pages 265–272, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.
- [16] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 71–78, New York, NY, USA, 1995. ACM.
- [17] Paul M. Isaacs and Michael F. Cohen. Controlling dynamic simulation with kinematic constraints. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 215–224, New York, NY, USA, 1987. ACM.
- [18] Duane Knudson. *Fundamentals of Biomechanics*. Springer, 2007.
- [19] Taku Komura, Howard Leung, and James Kuffner. Animating reactive motions for biped locomotion. In *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 32–40, New York, NY, USA, 2004. ACM.
- [20] Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. Limit cycle control and its

- application to the animation of balancing and walking. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 155–162, New York, NY, USA, 1996. ACM.
- [21] Hironori Mitake, Kazuyuki Asano, Takafumi Aoki, Marc Salvati, Makoto Sato, and Shoichi Hasegawa. Physics-driven multi dimensional keyframe animation for artist-directable interactive character. *Comput. Graph. Forum*, 28(2):279–287, 2009.
- [22] Uldarico Muico, Yongjoon Lee, Jovan Popović, and Zoran Popović. Contact-aware nonlinear control of dynamic characters. *ACM Trans. Graph.*, 28(3):1–9, 2009.
- [23] ODE. Open dynamics engine. <http://www.ode.org>.
- [24] Masaki Oshita and Akifumi Makinouchi. A dynamic motion control technique for human-like articulated figures. *Comput. Graph. Forum*, 20(3), 2001.
- [25] Zoran Popović and Andrew Witkin. Physically based motion transformation. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [26] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. *SIGGRAPH Comput. Graph.*, 25(4):349–358, 1991.
- [27] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.*, 23(3):514–521, 2004.
- [28] Ken Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254, 1985.
- [29] Kwang Won Sok, Manmyung Kim, and Jehee Lee. Simulating biped behaviors from human motion data. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 107, New York, NY, USA, 2007. ACM.
- [30] Kim Soohwan, Kim Minkyung, and Park Minje. Combining motion capture data with pd controllers for human-like animations. *SICE-ICASE International Joint Conference*, 0:904–907, 2006.
- [31] M. Van de Panne and E. Fiume. Virtual wind-up toys. In *Proceedings of Graphics Interface 94*, 1994.
- [32] Andrew Witkin and Michael Kass. Space-time constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 159–168, New York, NY, USA, 1988. ACM.
- [33] Pawel Wrotek, Odest Chadwicke Jenkins, and Morgan McGuire. Dynamo: dynamic, data-driven character control with adjustable balance. In *Sandbox '06: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, pages 61–70, New York, NY, USA, 2006. ACM.
- [34] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: simple biped locomotion control. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 105, New York, NY, USA, 2007. ACM.
- [35] Victor Brian Zordan and Jessica K. Hodgins. Motion capture-driven simulations that hit and react. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 89–96, New York, NY, USA, 2002. ACM.
- [36] Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. Dynamic response for motion capture animation. *ACM Trans. Graph.*, 24(3):697–701, 2005.